RAPPORT FINAL

Projet d'authentification unique

Projet réalisé dans le cadre du cours
INF4173 - Projet synthèse
Présenté à
Monsieur Michal Iglewski et Monsieur Stéphane Gagnon

Par
Michel Harper

Universite du Québec en Outaouais
Gatineau (Québec)

Table des matières :

Problématique

Les portails GagnonTech & IndusProd sont la propriété de l'entreprise Innovations Intracubator Inc (http://www.intracubator.com). Il s'agit d'un petit groupe conseil en TI incorporé depuis 1997 à Montréal. Il est dirigé par le Dr. Stéphane Gagnon, un nouveau professeur au DSA à l'UQO.

Ces 2 portails en développement visent à offrir les services suivant :

- GagnonTech : service intégré de formation sur mesure, d'édition de livres et revues, d'organisation de conférences, de gestion de regroupements/associations, de services de sondages et recherches, et de placement pour les professionnels.

- IndusProd : service intégré de e-commerce pour les vendeurs et acheteurs professionnels inter-entreprise (business to business, B2B), offrant l'opportunité d'ajouter des produits au catalogue du site, d'acheter des produits listés par d'autres usagers, et de gérer les processus de relation client, d'approvisionnement, et de gestion comptable (Enterprise Resource Planning, ERP).

Chacun de ces services est déjà implantés par une ou plusieures solutions open sources bien connus.

La difficultée est que chacune de ces solutions est une application en elle-même avec ses propre déploiement, base de donnée, liste d'usagers et liste de droits. Ceci rend la tâche complexe autant aux administrateurs du portail qu'aux utilisateurs, qui seraient obligés de s'authentifier à un bon nombre d'applications à chaque visite sur le portail. Les données se trouvant sur ces portails sont privées, on ne peut donc pas simplement éliminer l'étape d'authentification.

Méthodologie

La solution proposé lors de la phase d'analyse, précédant mon intégration au projet, était d'implémenter un module d'authentification unique permettant de s'authentifier et d'administrer les utilisateurs de chacune des solutions faisant partie du portail à partir du portail principal du site.

Une approche suivant le Rational Unified Process à été choisie afin d'amener ce projet à terme. Les différentes étapes qui ont servit à structurer le projet sont :

1. **Évaluation :** Évaluation des solutions open source retenues selon la grille des besoins et fonctionnalités désirées.
2. **Architecture :** Identification des points d'intégration possibles et design d'une architecture pour la solution pouvant servir de point d'accès commun (single-sign-on) entre toutes les applications retenues.
3. **Design :** Design des processus et spécification de l'environnement de gestion intégré de ce point d'accès (chaque portail a besoin d'un admin panel pour gérer leurs utilisateurs respectifs, à travers toutes les applications retenues).
4. **Programmation :** Programmation de la plateforme d'intégration et test d'intégrité de la sécurité, des données, des profils d'usagers, des sessions, des processus et des transactions intra- et inter-applications.
5. **Test :** Consultation avec les usagers/collaborateurs pour compléter les premiers tests betas.

Chacune des solutions faisant partie du portail est disponible sous une licence open source et le superviseur tien à ce qu'elles le restent. Ces solutions sont développée en PHP et supportée par une base de donnée MySQL. L'utilisation de la plateforme Linux Apache MySQL PHP (LAMP) était donc de mise.

Finalement un site de discussions a été mis en-ligne afin de faciliter les échanges entre les personnes impliquées dans le projet ainsi que suivre le développement de la solution.

Introduction aux applications

Joomla! : Il s'agit en fait d'un système de gestion de contenu. Il permet donc de gérer l'accès et l'affichage d'une grande quantité de contenu de toutes les sortes (texte, images, calendrier). Joomla! est aussi extrêmement flexible grâce à son système de composantes et de modules, qui permet d'ajouter ou d'enlever des capacité au portail. Il est utilisé un peu partout dans le monde par les petits et les gros organisme, pour plus d'information voir http://www.joomla.org/. L'administrateur peut aussi modifier la disposition du contenu. Joomla! est le portail principal utilisé sur les 2 sites cibles et c'est de cette application que l'on souhaite gérer toutes les autres.

Moodle : C'est un système de gestion de cours. Un administrateur peut donc créer des cours et y assigner un ou des professeurs. Les professeurs peuvent rendrent disponibles des plans de cours, des forums de discussion et toute sorte de documentation utile. Les élèves peuvent s'inscrire à des cours, suivre le plan cours après cours et discuter avec le professeur au travers les forums, voir http://moodle.org/. C'est l'application qui à été priorisée par le superviseurafin que l'intégration soit fonctionelle lors de notre remise de projet.

Analyse

La premiere partie de l'analyse était d'analyser l'implémentation de Joomla! et des 15 autres applications choisie pour faire partie du portail, afin de situer plus précisément le code relatif à la gestion des utilisateurs.

Avec ces connaissances en main, nous avons pu discuter de différentes possibilités d'implémentation d'un module d'authentification unique. Plusieures solutions fûrent proposée :

1. Centrée sur les **objets externes** = Joomla appelle les classes des autres applications.
2. Centrée sur le **SQL** = Joomla intègre dans son code les commandes SQL des autres applications.
3. Centrée sur un **nouvel objet** = Joomla posède un nouvel objet qui exécutera de manière itérative les tâches sur plusieurs applications externes.
4. Centrée sur le **composant** = Joomla aura un nouveau composant qui intègrera en entier les fonctions des autres applications.

Nous avons aussi profité de la rencontre pour séparer les tâches de développement afin de pouvoir travailler en parallèle, quatre concept furent choisi qui existait dans chacune des application : l'enregistrement d'un nouvel usager, l'édition des données d'une usager, l'authentification aux systèmes et la prise en charge de variables de session. Je reçu l'édition d'usagers.

Il devint aussi évident que l'implémentation d'une solution intégrant les 15 applications serait une trop

grosse charge de travail pour le temps alloué au projet. Nous avons donc décidé de viser principalement 3 applications.

Nous avons donc analysé chacune des application afin de déterminer quelle solution d'implémentation serait la plus approprié. La solution « composant » est ressortie comme beaucoup trop complexe pour l'utilisation minimale dont nous avions besoin. Les solutions « objets externes » et « SQL » fûrent mise de côté parce qu'elles nécessitait de modifier grandement le code de Joomla! Ce qui rendrait l'implémentation plus difficile, sans parler de la mauvaise pratique de modularisation que cela constitue. Nous avons, donc, décidé d'implémenter un nouvel objet qui pourrait être appelé aux bons endroits dans Joomla! et qui effectuera les changements nécessaires dans les applications cibles par l'appel de commandes SQL.

Solution

L'élaboration de la solution s'est avérée plutôt simple, le plus gros du travail étant de comprendre le code élaboré par d'autres programeurs et de trouver un modèle qui fonctionnera pour au moins la majorité des 15 applications. Une fois le bon emplacement trouvé dans Joomla! j'ai tout simplement ajouté une ligne qui fait appel à un nouveau fichier que j'ai créé et que l'on peut simplement copier dans le répertoire principal de Joomla!. Ensuite j'ai dû analyser le code de Moodle afin de trouver la commande SQL exacte qui est lancée lorsqu'un utilisateur modifie ses coordonnées. J'ai aussi noté les restrictions appliqués aux données lors de la soumissions dans Moodle, afin de s'assurer que les donnés fournie par Joomla! sont correcte. En effet les applications utilisent quand même leur base de donnée respective et doivent recevoir de l'information de celles-çi, on ne doit pas corrompre les donnés en fournissant des informations incorrectes. Je me suis aussi assuré que les champs obligatoires des deux applications étaient rempli, on assume qu'il y a une raison valide pour chaque champs obligatoire. Il y a une différence notable entre les deux : Joomla! traite le vrai nom de l'utilisateur avec un seul champs qui contient son nom et prénom, Moodle possède un champs pour nom et prénom, j'ai donc du programer une séparation du prénom au premier caractère d'espacement.

Afin d'être plus facile à modifier et plus dynamique, le module utilise une base de donnée MySQL qui contient une liste des actions possible pour un utilisateur, ajouter et modifier pour le moment, une liste des applications supportées et une ou plusieurs entrées « commande » qui seront exécutées par le module. La base de donnée contient aussi les informations nécessaire pour se connecter à la base de donnés utilisée par chaque application cible ainsi que le nom de la table ou sont sauvegardées les informations sur les usagers. La meilleure façon que j'ai trouvé d'enregistrer la commande à exécuter par le module est en fait de conserver une liste associative du nom du champs soumis dans Joomla! vers le nom du champs dans la table conservant les usagers dans l'application cible. Ces informations sont ordonnés à la façon d'une chaîne d'arguments d'une addresse web ex : « name=uname&pass=password ». J'ai ensuite pu utiliser la fonction parse_str de php pour obtenir chaque valeur indépendament des autres e tconstruire la requête SQL.

Le module n'assume pas que les ID interne des utilisateurs dans chaque base de données sont les même, la plupart du temps ils s'agit de valeurs auto-incrémenté, qui ne peuvent être changé sans corrompre la base de donnée, et il était spécifié qu'un utilisateur pouvait avoir accès ou pas à une ou plusieurs applications, donc il est inutile de créer un utilisateur dans toutes les bases de données à chaque fois. J'ai plutôt utilisé le nom d'utilisateur pour repérer le bon utilisateur à changer, il s'agit en effet d'une valeur unique dans toutes les applications et il est pratiquement nécessaire, sinon très utile d'utiliser le même nom d'utilisateur au travers chacune des application. Le cas ou l'utilisateur veut changer sont nom d'utilisateur n'est pas problématique puisque j'utilise en fait le nom d'utilisateur courant(ancien)

pour trouver le bon champs.

Afin d'assurer une bonne cohésion des donnés au travers toutes les applications le code inséré dans Joomla! s'assure que le module s'est exécuté sans erreur avant de sauvegarder définitevemt les données.

Il était important pour notre supperviseur d'avoir une idée de l'effet de notre module sur les différentes applications lorsqu'il sera déployé, la base de donnée du module contient donc une table « Journal » qui contient la commande SQL exacte qui a été appelé annexé d'une date et heure. Il pourrait être intéressant dans une implémentation future de sauvegarder les informations qui sont remplacés par la commande, afin de faciliter un rollback si cela devien nécessaire.

Continuité

L'implementation de la synchronisation de l'édition des informations de l'usager a été un succès. Il reste par contre beaucoup de travail à faire avant le déploiement de la solution complète.

Tout d'abord, chaque équipe a travaillé sur ce projet chacun de son coté, tout en ayant une architecture semblable, les projets ne sont pas encore assemblé afin qu'un utilisateur puisse voyager entre Moodle et Joomla! de façon transparente.

Puisque nous nous sommes finalement concentré sur une seule application, il reste encore une quinzaine d'applications qui doivent être implémentée. Ce travail devrait être simple, nous avons déjà localisé les emplacements où trouver les informations nécessaires, il s'agit en fait que de faire de l'entrée de données.

Ensuite, il y a deux aspect que me sont apparu lors de l'analyse qui n'ont pas été mentionné spécifiquement par le superviseur et qui pourraient surement être utile à long terme. Premièrement, il s'agit de l'implémentation de fonctions permettant à un administrateur de modifier les données d'un utilisateur pour toutes les applications. En effet, il est possible pour un administrateur du site de modifier les informations sur un utilisateur pour chacun des sites, mais pas ensemble. Deuxièment, il serait intéressant de soit désactiver les modifications des données de l'utilisateur dans l'application cible ou de créer un script qui effectuerait les changements inverse, la première option semble plus plausible. Ces deux aspect permettraient de s'assurer de la cohésion des données peu importe ce que font les utilisateur/administrateurs dans l'application.

Finalement, une période de tests sera définitevement nécessaire avant de remettre notre solution aux mains du public.

Conclusion

Pour conclure, ce projet m'a permis d'en apprendre beaucoup sur les applications open source, beaucoup de solutions sont déjà disponibles gratuitement et implémentent des solutions a de multiples problèmes en tout ou en partie à une qualité semblable à des logiciels commerciaux. Ce fût aussi une excellente experience d'analyse d'un logiciel plus grand que ce qu'un seul programmeur peut faire. J'ai aussi appris à travailler avec de nouveaux logiciels dont XAMPP, qui est une implémentation de LAMP que l'on peut tout simplement télécharger sur sont ordinateur et commencer à travailler dans un environnement ouvert.

J'ai aussi appris qu'il pouvait être difficile de comprendre les besoins d'un client simplement par ce qu'il vous dit lui-même, il est important de garder le contact et de poser des question sur les points mitigé et de se réorienter si s'est nécessaire. Finalement il a aussi été complexe de travailler tout en prennant en considération qu'il y a plusieurs autres personnes qui travaillent sur le même projet et on ne doit pas modifier des donnés qui pourrait leur être utile.

Bibliographie

Rationnal Unified Process par IBM

http://www-306.ibm.com/software/rational/uml/

http://www.joomla.org/

http://moodle.org/

http://help.joomla.org/images/User_manual/user_manual_v1%200%201_10%2021%2006.pdf

http://www.apachefriends.org/en/xampp-windows.html

Annexe A (SQL)

Application

ApplicationID (*)
ApplicationName
ApplicationSourceSite
ApplicationDbType
ApplicationDbHostname
ApplicationDbDatabaseName
ApplicationDbUsername
ApplicationDbPassword

Action

ActionID (*)
ActionName

SQLCommand

SQLCommandID (*)
ApplicationID (fk)
ActionID (fk)
CommandText
CommandOrder

CommandJournal

CommandJournalID (*)
ExecutionDateTime
Username
TargetApplication
CommandText
ScriptName

Annexe B (Code)

external_app.php :

```
<?php
  function updateData($orig_username, $new) {
    $mysql_host = 'localhost';
    $mysql_user = 'root';
    $mysql_password = '';
    $mysql_db = 'projetsynthese';
    $action = 'update';

    $names = explode(" ", $new->name);
    $firstname = $names[0];
    $lastname = "";
    foreach ( $names as $key => $value) {
      if ($key != 0) {
        $lastname .= $value . " ";
      }
    }
    $lastname = rtrim($lastname);

    // Connecting, selecting database
    $link = mysql_connect($mysql_host, $mysql_user, $mysql_password)
       or die('Could not connect1: ' . mysql_error());
    mysql_select_db($mysql_db) or die('Could not select database');

    // Performing SQL query
    $sql = "SELECT application.ApplicationDbHostname,
                   application.ApplicationDbUsername,
                   application.ApplicationDbPassword,
                   application.ApplicationDbDatabaseName,
                   sqlcommand.CommandText,
                   application.ApplicationID
            FROM `sqlcommand`
              INNER JOIN (application, action)
              ON (application.ApplicationID = sqlcommand.ApplicationID
              AND action.ActionID = sqlcommand.ActionID)
            WHERE action.ActionName = '" . $action . "'
            ORDER BY application.ApplicationID, sqlcommand.CommandOrder ASC";

    $result = mysql_query($sql) or die('Query failed: ' . mysql_error());

    while ($line = mysql_fetch_array($result, MYSQL_NUM)) {
      $innerlink = mysql_connect($line[0], $line[1], $line[2])
        or die('Could not connect2: ' . mysql_error());
      mysql_select_db($line[3]) or die('Could not select database');

      //Returns : $arr[0]=table_name
      //          $arr[1]=username
```

```
    //          $arr[2]=password
    //          $arr[3]=firstname
    //          $arr[4]=lastname
    //          $arr[5]=email
    //          $arr[6]=lastlogin
    //          $arr[7]=id
    parse_str($line[4]);

    $innersql = "SELECT " . $arr[7] . " FROM " . $arr[0] .
                " WHERE " . $arr[1] . " = '" . $orig_username .
                "' LIMIT 1";

    $innerresult = mysql_query($innersql) or die('Query failed: ' . mysql_error() . $innersql);
    $innerline = mysql_fetch_array($innerresult, MYSQL_NUM)
      or die('Could not connect3: ' . mysql_error() . $innersql);
    $to_update_id = $innerline[0];

    $innersql = "UPDATE " . $arr[0] .
                " SET " . $arr[1] . " = '" . $new->username .
                "', " . $arr[2] . " = '" . $new->password .
                "', " . $arr[3] . " = '" . $firstname .
                "', " . $arr[4] . " = '" . $lastname .
                "', " . $arr[5] . " = '" . $new->email .
                "', " . $arr[6] . " = '" . $new->lastvisitDate .
                "' WHERE " . $arr[7] . " = " . $to_update_id .
                " LIMIT 1";
    mysql_query($innersql) or die('Error updating other databases');

    mysql_free_result($innerresult);
    mysql_close($innerlink);

    $journallink = mysql_connect($mysql_host, $mysql_user, $mysql_password)
      or die('Could not connect4: ' . mysql_error());
    mysql_select_db($mysql_db) or die('Could not select database');
    $innersql = str_replace("'", "\'", $innersql);
    $sql_journal = "INSERT INTO commandjournal
                    SET Username = '" . $line[1] .
                    "', TargetApplicationID = " . $line[5] .
                    ", CommandText = '" . $innersql .
                    "', ScriptName = 'external_app.php:updateData($orig_username, $new)'";

    mysql_query($sql_journal) or die('Error updating journal table' . $sql_journal);

    mysql_close($journallink);
  }
  mysql_free_result($result);
  mysql_close($link);
  return true;
}
```